Selecting Data

Duration : 8 hrs

© 2007, UCSC

Detailed Syllabus

4.3 Selecting Data

4.3.1 Queries: SELECT Statement.

4.3.2 Single Table: all columns (*), selecting specific columns (RA project operation), unique values (DISTINCT), Executing multiple statements (;), WHERE clause (RA select operation), Including or excluding rows (=, !=), Relational Operators (=, !=, >, >=, <, <=), Identifying Null values (IS NULL), Where clause keywords (AND, OR, [NOT] BETWEEN, [NOT] IN, IS [NOT] NULL, [NOT] LIKE, ORDER BY, Arithmetic Operators (+, -, *, /), Expressions, Display Labels, Aggregate Functions: COUNT, SUM, AVG, MAX, MIN, GROUP BY, HAVING.

4.3.3 Multiple Table: RA join and product operations, Natural Join, Multiple Table Joins, Aliases for table names, Outer Join, UNION.

4.3.4 Functions: Arithmetic (ROUND, TRUNC), String (TO_CHAR, UPPER, LOWER, Sub strings, Concatenation, TRIM), Date and Time (DAY, MONTH, YEAR, DATE, CURRENT).

4.3.5 Sub queries: Nested Select Statement, Values returned by sub queries (single value, a list of values), EXISTS, Correlated nested queries.

© 2007, UCSC

Referential Integrity

SQL data definition for defining referential integrity constraints

Parent table:

```
CREATE TABLE DEPARTMENT
(DEPT-NO  CHAR(3),
other column definitions
PRIMARY KEY (DEPT-NO) );
```

Dependent table:

```
CREATE TABLE EMPLOYEE
    ( EMP-NO          CHAR(5),
      DEPT-NO         CHAR(3)
      other column definitions
      PRIMARY KEY (EMP-NO),
      FOREIGN KEY DEPT-N-FK (DEPT-NO)
            REFERENCES DEPARTMENT
        ON DELETE SET NULL) );
```

© 2007, UCSC

Defining referential integrity rules in the SQL DDL is known as ***declarative*** referential integrity

Declarative referential integrity simplifies application programming and enables enforcement at the database server level, eliminating the possibility of programming errors

User Defined Integrity

User defined integrity constraints can be enforced by the database server using ***triggers*** and ***stored procedures***.

© 2007, UCSC

Triggers and stored procedures are user written routines which are stored and executed under the control of the database server.

They are often coded in proprietary procedural extensions to SQL, e.g. Sybase's Transact SQL or Oracle's PL/SQL.

SQL for Data Manipulation

Manipulation

SQL allows a user or an application program to update the database by adding new data, removing old data, and modifying previously stored data.

© 2007, UCSC

Retrieval

SQL allows a user or an application program to retrieve stored data from the database and use it.

- **Most Commonly Used Commands**
  - SELECT INSERT
  - UPDATE DELETE

© 2007, UCSC

SQL for Data Manipulation

-High-level Language for data manipulation

-It does not require predefined navigation path

-It does not require knowledge of any key items

-It is uniform language for end-users and programmers

-It operates on one or more tables based on set theory, not on a record at a time.

© 2007, UCSC

Command: **SELECT**

Function

– *Retrieves data from one or more rows. Every SELECT statement produces a table of query results containing one or more columns and zero or more rows.*

SELECT {[ALL, DISTINCT]}

[(*select-item*,), i]

- FROM (*table specification*,)
- {WHERE (*search condition*)}
- {GROUP BY (*group-column*,)}
- {HAVING (*search condition*)}
- {ORDER BY (*sort specification*,)}

© 2007, UCSC

Project Selected Columns

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

SELECT E-No, E-Name
FROM Employee ;

Employee Names

E-No	E-Name
179	Silva
857	Perera
342	Dias

SELECT E-No, E-Name

FROM Employee
ORDER BY E-Name ;

Employee Names

E-No	E-Name
342	Dias
857	Perera
179	Silva

© 2007, UCSC

Restrict Rows

Sales Employee

E-No	E-Name	D-No
179	Silva	7
342	Dias	7

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

```
SELECT *
FROM   Employee
WHERE D-No = '7' ;
```

```
SELECT E-No, E-Name
FROM    Employee
WHERE D-No = '7' ;
```

Sales Employee

E-No	E-Name
179	Silva
342	Dias

Restrict Rows and Project Columns

© 2007, UCSC

EquiJoin

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

Department

D-No	D-Name	M-No
4	Finance	857
7	Sales	179

Emp-Info

E-No	E-Name	D-No	D-No	D-Name	M-No
179	Silva	7	7	Sales	179
857	Perera	4	4	Finance	857
342	Dias	7	7	Sales	179

```
SELECT    Employee.*, Department.*
FROM      Employee, Department
WHERE     Employee.D-No = Department.D-No ;
```

```
SELECT    E.*, D.*
FROM      Employee E, Department D
WHERE     E.D-No = D.D-No ;
```

© 2007, UCSC

Inner Join

```
SELECT E.*, D.*
FROM Employee E
INNER JOIN Department D ON E.D-No = D.D-No;
```

Outer Joins: Left, Right, Full

```
SELECT E.*, D.*
FROM Employee E LEFT OUTER JOIN Department D ON E.D-No = D.D-No;
```

© 2007, UCSC

Department

D-No	D-Name	M-No
2	Finance	850
7	Sales	179

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

Emp-Info

E-No	E-Name	D-No	D-No	D-Name	M-No
179	Silva	7	7	Sales	179
857	Perera	4	Null	Null	Null
342	Dias	7	7	Sales	179

© 2007, UCSC

Emp-Info

E-No	E-Name	D-No	D-No	D-Name	M-No
Null	Null	Null	2	Finance	850
179	Silva	7	7	Sales	179
342	Dias	7	7	Sales	179

Right Outer Join

SELECT E.*, D.*
FROM Employee E **RIGHT OUTER JOIN** Department D
ON E.D-No = D.D-No;

Full Outer Join

SELECT E.*, D.*
FROM Employee E **FULL OUTER JOIN** Department D
ON E.D-No = D.D-No;

© 2007, UCSC

Cartesian Product

Department

D-No	D-Name	M-No
4	Finance	857
7	Sales	179

Employee

E-No	E-Name	D-No
179	Silva	7
857	Perera	4
342	Dias	7

Emp-Info

E-No	E-Name	D-No	D-No	D-Name	M-No
179	Silva	7	4	Finance	857
857	Perera	4	4	Finance	857
342	Dias	7	4	Finance	857
179	Silva	7	7	Sales	179
857	Perera	4	7	Sales	179
342	Dias	7	7	Sales	179

SELECT

E.*, D.*

FROM

Employee E, Department D

© 2007, UCSC

SQL Data Retrieval

Basic Search Conditions

Comparison

- Equal to =
- Not equal to != or <> or ^=
- Less than to <
- Less than or equal to <=
- Greater than to >
- Greater than or equal to >=

© 2007, UCSC

SQL Data Retrieval

Basic Search Conditions (cont'd)

- Range ([NOT] BETWEEN)
  - *expres-1* [NOT] BETWEEN *expres-2* AND *expres- 3*
  - *Example:* WEIGHT **BETWEEN** 50 AND 60
- Set Membership ([NOT] IN)
  - Example 1: WHERE Emp_No IN ('E1', 'E2', 'E3')
  - Example 2: WHERE Emp_No IN (Select Emp_No FROM Employee WHERE Dept_No='7')

© 2007, UCSC

Basic Search Conditions (cont'd) **:**

- Pattern Matching ([NOT] LIKE)
  - *expres-1* [NOT] LIKE {*special-register* | *host-variable* | *string-constant*}
  - *Example:* WHERE Proj_Name **LIKE** "INFORM%"
- Null Value (IS [NOT] NULL)
  - *Example:* WHERE Proj_Name **IS** NOT NULL

© 2007, UCSC

Compound Search Conditions

AND, OR and NOT

Example:

WHERE Proj_Name **LIKE** 'INFORM%' AND Emp_Name = 'DIAS'

SQL Query Features

- Summary Queries
  - *Summarize data from the database. In general, summary queries use SQL functions to collapse a column of data values into a single value that summarizes the column. (AVG, MIN, MAX, SUM, COUNT..)*
- *Sub-Queries*
  - *Use the results of one query to help define another query*

© 2007, UCSC

Summarising Data

SELECT COUNT(*)
FROM Employee

Employee

E-No	Job	Salary	D-No
179	Manager	20000	10
857	Clerk	8000	10
342	Clerk	9000	20
477	Manager	15000	30
432	Clerk	10000	30

Count(*)
5

SELECT AVG(Salary)
FROM Employee

AVG(Salary)
12400

© 2007, UCSC

SELECT STATEMENT
May also contain
[GROUP BY [HAVING] ORDER BY]

GROUP BY

A result of a previous specified clause is grouped using the group by clause.

e.g.

```
SELECT      d-no, AVG(salary)
FROM        employee
GROUP BY    d-no
```

Employee

E-No	Job	Salary	D-No
179	Manager	20000	10
857	Clerk	8000	10
342	Clerk	9000	20
477	Manager	15000	30
432	Clerk	10000	30

D-No	AVG(Salary)
10	14,000
20	9,000
30	12,500

© 2007, UCSC

HAVING

Used for select groups that meet specified conditions.

Always used with GROUP BY clause.

```
SELECT      d-no, AVG(salary)
FORM        employee
GROUP BY    d-no
HAVING      AVG(salary)>12000
```

Employee

E-No	Job	Salary	D-No
179	Manager	20000	10
857	Clerk	8000	10
342	Clerk	9000	20
477	Manager	15000	30
432	Clerk	10000	30

D-No	AVG(Salary)
10	14,000
30	12,500

© 2007, UCSC

Nested Queries

A sub query is SELECT statement that nest inside the WHERE clause of another SELECT statement. The results are need in solving the main query.
Get a list of all suppliers supplying part P2.

```
SELECT sname
FROM supplier
WHERE sno IN
(SELECT sno FROM supply WHERE pno = 'P2');
```

```
SELECT sname
FROM supplier, supply
WHERE supplier.sno = supply.sno and pno = 'P2';
```

```
SELECT ename , salary
FROM employee
WHERE salary = (SELECT MIN (salary) FROM employee)
```

© 2007, UCSC

Nested Queries Contd.

Sub queries with EXISTS

e.g. find all publishers who publish business books

```
SELECT    DISTINCT    pub_name
FROM publishers
WHERE EXISTS
(SELECT * FROM title
 WHERE pub_id = publishers.pub_id and type = “business”)
```

DISTINCT – will remove multiple occurrences

© 2007, UCSC